Slotted Wavelength Scheduling for Bulk Transfers in Research Networks

Zhe Wang, Sanjay Ranka and Ye Xia

***Abstract*—The advancement of optical network technologies has enabled data-intensive e-science collaborations, which often require the transfer of large files with predictable performance. To support such applications, we design and evaluate two algorithms for scheduling time-constrained bulk transfers on wavelength-based optical research networks. The first one seeks to maximize the network throughput while maintaining a level of fairness among the jobs. The second algorithm works in a overloaded network and serves as an alternative to the first algorithm. It seeks to extend the end times by the smallest possible proportion and complete all the jobs by the extended end times. The main challenge is that the underlying problems are integer optimization problems for wavelength assignment, which have no known fast optimal solutions. We present a heuristic sub-algorithm called LPDAR, which converts fractional solutions from linear programming into integer solutions. LPDAR is the key component used in both aforementioned algorithms. Evaluation shows that LPDAR leads to very good algorithms with a performance level and speed both comparable to those of the LP fractional solutions.**

***Index Terms*—Wavelength Scheduling, Advance Reservation, Admission Control, Bulk Data Transfer, E-science**

I. Introduction

The advance of communication, networking and computing technologies is dramatically changing the ways how scientific research is conducted. A new term, *e-science*, has emerged to describe the "large-scale science carried out through distributed global collaborations enabled by networks, requiring access to very large scale data collections, computing resources, and high-performance visualization" [1]. Well-quoted e-science (and the related grid computing [2]) examples include high-energy nuclear physics (HEP), radio astronomy, geoscience and climate studies. To support e-science activities, a new generation of high-speed research and education networks have been developed. These include Internet2 [3], the Department of Energy's ESnet [4], National Lambda Rail [5], CA*net4 [6] in Canada, and the pan-Europe GÉANT2 [7]. A large fraction of all data traffic supporting U.S. science is carried by ESnet, Internet2, and National Lambda Rail [8].

The need for transporting large volume of data in e-science has been well-argued [9], [10]. For instance, the HEP data is expected to grow from the current petabytes (PB) (10^{15}) to exabytes (10^{18}) by 2012 to 2015. In addition, e-scientists desire schedulable network services to support predicable work processes [11]. An important type of schedulable services is large data transfers with start and end time requirement. In such a service, each data-transfer job request can be made in advance and can specify a start time and an end time. If it has sufficient resources, the network guarantees that it will begin the data transfer after the requested start time and finish it before the requested end time. This paper presents *admission control* (AC) and *scheduling* algorithms for supporting large data transfers in research networks with the start and end time requirement. The targeted environment is wavelength-switched networks, and hence, the AC/scheduling problems are particular kinds of wavelength assignment problems, which in general are very difficult to solve. Our focus is on developing high-performance and fast algorithms. The results should complement other protocol, architecture or infrastructure projects in support of e-science and grid computing [3], [4], [7], [12], [13], [14], [15], [16].

The paper adopts the optimization-based paradigm proposed by several earlier papers [17], [18], [19], [20], [21]. Specifically, the AC and scheduling decisions are made by the network controller that has a global view of the network and the job requests. The controller solves optimal resource allocation problems with start and end time constraints in making these decisions. The main benefit of this approach, as demonstrated in [20], [21], is greatly improved network resource efficiency.

This paper continues the earlier work reported in [20], [21] and the distinctions are as follows. In the earlier papers, it is assumed that the bandwidth of every link is infinitely divisible; this is a useful assumption to model an all-router packet network. Linear programming problems were formulated where the decision variables were real-valued. However, the next-generation research networks will be a mixture of IP packet networks, traditional optical circuit-switched networks (e.g., SONET), and wavelength-switched networks [14]. The algorithms in [20], [21] can handle the former two types of networks. This paper extends the earlier work to wavelength-switched optical networks. For this type of networks, the optimization problem involves integer variables, which represent the numbers of wavelengths assigned to the data transfer jobs. The resulting integer programming problem typically takes prohibitively long computation time to solve. One of the main contributions of the current paper is that it gives novel solution techniques that require far less computation time at the expense of only small loss of optimality.

Recently, some other authors have begun to study AC

The authors are with the Computer and Information Science and Engineering Department, University of Florida, Gainesville, FL.

Ye Xia is the corresponding author. Email: yx1@cise.ufl.edu, Phone: 352-392-2714, Fax:352-392-2714

This work was supported in part by the National Science Foundation (NSF) under Grant PHY 0312038 and PHY 0622423. Any findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF. The authors would like to thank Rick Cavanaugh and Paul Avery for several discussions and insights.

and scheduling algorithms for bulk transfers with advance reservations [22], [17], [23], [19], [24], [25]. Compared with these earlier studies, our work distinguishes itself for using the optimization-based formulation or for the degree of scheduling flexibility that our formulation allows. Our approach will translate into much greater resource efficiency. Secondly, few prior studies have considered the time-constrained bulk transfer problem on a wavelength-switched network under an optimization framework. Thirdly, our focus is on the overloaded situation where the network does not have sufficient resources to satisfy all the requests. Instead of just rejecting some job requests, we also consider a negotiation-based approach where the users may modify the job parameters and and re-submit the modified requests. The main content of the paper describes two algorithms: One allows reducing the job sizes; the other allows extending the end times. Finally, our formulations incorporate the fairness issues while dealing with multiple jobs.

The problems of the paper are also different from typical wavelength assignment problems (e.g., see [26], [27]), where the objective is to establish long-lasting light paths. Here, the time dimension is important: First, bulk transfer is elastic and can take time-varying wavelength assignments; second, the start and end time constraints call for very different formulations of the scheduling problems.

The rest of the paper is organized as follows. Section II describes the two algorithms we developed for AC/scheduling. Section III shows the performance evaluations of the proposed algorithms. The related work is shown in Section IV. The conclusion is drawn in Section V.

II. Admission Control and Scheduling Algorithms

A. System Overview

We first present the system model and a high-level overview of the AC and scheduling framework proposed in [21]. This paper extends the framework to wavelength-switched networks by providing two AC/scheduling algorithms for bulk transfer jobs with time constraints and integrality constraints.

The network is represented by a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. The capacity of a link (an edge) $e \in E$ is denoted by C_e. Since we are dealing with a wavelength-switched network, C_e is the total number of wavelengths of link e. Job requests arrive at the network following a random process. Each job request i is a 6-tuple $(A_i, s_i, d_i, D_i, S_i, E_i)$, where A_i is the arrival time of the request, s_i and d_i are the source and destination nodes, respectively, D_i is the size of the job, S_i and E_i are the requested start time and end times, where $A_i \leq S_i \leq E_i$. In words, request i, which is made at time $t = A_i$, asks the network to transfer a file of size D_i from node s_i to node d_i over the time interval $[S_i, E_i]$.

In our framework, the network resource is managed by a small number of *network controllers* (NC), which control job admission, bandwidth (wavelength) allocation and setup of the allocated resources. One advantage of the approach is that resource reservation and allocation decisions are based on a global view of the network and on all the job requests. It is possible to manage the network resources as a whole and make tradeoffs among all the jobs in the network. The result is greatly improved efficiency in network resource utilization. Different from the public Internet, research networks typically have far fewer than 1000 core nodes in the backbone. Hence, it possible to use a centralized approach for keeping track of the global information about the whole network and all the jobs. Virtually all current and planned research networks have adopted the approach of using centralized controllers for service/resource reservation and scheduling. However, none formulate and solve optimization problems for making the resource allocation decisions. Our proposal to use a small number of controllers and have them solve optimization problems periodically has been shown to be effective enough for router-based research networks [20], [21]. This paper will present solution techniques that are fast enough for wavelength-switched networks. If needed, the scalability can be further enhanced by using a distributed hierarchy of controllers.

The time is divided into time slices (or time slots). Each job can be assigned different bandwidth by the controllers on different time slices. Each job can also take multiple paths simultaneously on each time slice, and these paths can be different on different time slices. The file transfer requests are submitted to the network controllers, which verify whether there are sufficient network resources to finish the jobs before the requested end times. If the resources are insufficient, some jobs may be rejected or the requested file sizes or end times may be re-negotiated. The process of deciding the admissibility of the job requests is called *admission control*. After AC, the network controllers will assign bandwidth to the admitted jobs on each time slice and on each allowed path. This assignment process is called *scheduling*. The scheduling stage may use the results from the AC stage. However, in general, the AC and scheduling stages may involve different optimization problems. AC and scheduling are conducted periodically every τ time units, where $\tau > 0$. More specifically, at time instances $k\tau$, $k = 1, 2, \ldots$, the controllers collect all the new requests that arrived on the interval $[(k-1)\tau, k\tau]$, make the AC decision first, and then, schedule the transfers of all jobs.

Both AC and scheduling must take into account the previous jobs, which have been admitted earlier but remain unfinished. The value of τ should be small enough so that new job requests can be checked for admission and scheduled as early as possible. However, it should be more than the computation time required for AC and scheduling. More details about this periodic AC/scheduling framework can be found in [21].

To extend the framework to wavelength-switched networks, the challenge is to find fast algorithms for handling the underlying integer optimization problems while maintaining high efficiency. The paper mainly describes two AC/scheduling algorithms that meet this challenge. The first one maximizes the throughput with end time guarantee. When end time guarantee is impossible due to excessive demand, three possible courses of actions can be taken: (i) Some jobs are rejected; (ii) the users agree to reduce the demand sizes and the network guarantees the end times for the modified demand

sizes; (iii) the users agree to relax the end times and the network guarantees the delivery of the entire jobs by the extended end times. Our first algorithm is suitable for action (ii). Its objective is to improve the network throughput while maintaining a level fairness among the jobs in the process of reducing the job sizes. Our second algorithm is suitable for action (iii). It finds the smallest (common) factor by which the end times should be extended so that all the jobs can be completed by the new end times. There is a simple algorithm for action (i) and we will not focus on it[1]. We leave developing more sophisticate algorithms for action (i) to future work.

B. Maximizing Throughput with End Time Guarantee

In this scheduling algorithm, our objective is to make efficient use of the network resources while maintaining fairness among the jobs. The resource efficiency will be measured in terms of a weighted throughput of the network. The end times of the jobs will be guaranteed by this algorithm, although this may require the users to reduce the demand sizes in an overloaded network. The core scheduling algorithm consists of solving two optimization problems in two stages.

- In stage 1, we pretend the bandwidth is infinitely divisible and formulate a linear programming problem. The optimal value Z^* will be called the maximum concurrent throughput of the network.
- In stage 2, we use Z^* to enforce fairness among the jobs while maximizing the weighted throughput, which is a better measure of the network resource efficiency than the maximum concurrent throughput.

The optimization problem in stage 2 is an integer program, which dominates the computation cost. We will present a good heuristic algorithm, LPDAR, to solve the stage-2 problem.

1) Stage 1 - Computing the Maximum Concurrent Throughput: The formulation in stage 1 is adapted from the well-known maximum concurrent flow (MCF) problem, which is a special type of network linear program. In the path-based formulation, the variables are the bandwidth assignments on the allowed paths of each job on every time slice, as well as the maximum concurrent throughput. For each file-transfer job, the formulation allows an explicitly defined collection of paths on every time slice and bandwidth reservations are done only on these paths.[2] We have found in [20] that a small number of paths per job (4 to 8 paths) is usually enough for achieving very good performance in the networks we tested.

Consider an arbitrary AC/scheduling instance $k\tau$, where k is a non-negative integer. The following variables should depend on k. For notational simplicity, we omit k in the notations. Let J be the set of job requests known to the system at time $k\tau$, waiting to be admitted and/or scheduled. Let $\mathcal{L}$ be the collection of future time slices that need to be considered. This collection is finite since it only needs to cover the largest of the requested end times of the known jobs. Let $P(s_i, d_i, j)$ be the set of allowed paths for job i on time slice j, where s_i is the source node and d_i is the destination node. Let $x_i(p, j)$ be the bandwidth assigned on path $p \in P(s_i, d_i, j)$ for job $i \in J$ on the time slice $j \in \mathcal{L}$. It is assumed to be constant on the entire slice. The problem formulation is given by:

Stage 1 (MCF)

$$\max \quad Z \tag{1}$$

$$\text{s.t.} \sum_{j\in\mathcal{L}} \sum_{p\in P(s_i,d_i,j)} x_i(p,j)\cdot LEN(j) = ZD_i, \forall i \in J \tag{2}$$

$$\sum_{i\in J} \sum_{\substack{p\in P(s_i,d_i,j)\\ p:e\in p}} x_i(p,j) \le C_e(j), \quad \forall e \in E, \forall j \in \mathcal{L} \tag{3}$$

$$x_i(p,j) = 0, \ \forall p \in P(s_i,d_i,j), j \le I(S_i) \text{ or } j > I(E_i), \quad \forall i \in J \tag{4}$$

$$x_i(p,j) \ge 0, \ \forall p \in P(s_i,d_i,j), \forall i \in J, \forall j \in \mathcal{L}. \tag{5}$$

In the stage-1 problem, we do not impose the integrality constraint; all variables are assumed to be real-valued. Z is called the concurrent throughput and the problem seeks to maximize the concurrent throughput. It asks: What is the maximum concurrent throughput Z^* such that, after every job size is scaled by Z^*, the link capacity constraints, as well as the start and end time constraints, are still satisfied for all time slices? Note that, if $Z^* < 1$, it is not possible to satisfy the deadline of all the jobs. However, if the file sizes are reduced by a common factor to Z^*D_i for all job i, then, the requests can all be satisfied. If $Z^* \ge 1$, then the file sizes can all be scaled up by the factor Z^* and still be handled by the network without any link capacity or deadline violation. We will loosely say the network is *overloaded* if Z^* is less than or equal to 1. If Z^* is greater than 1 by a non-trivial amount, we say the network is *underloaded*.

We next describe the formulation in more details.

- In (2), the left hand side is the total traffic scheduled to be transferred for job i. It is the sum of the flows assigned on all time slices and on all allowed paths for job i. Note that $LEN(j)$ is the length of slice j.
- Expression (3) says that the capacity constraints must be satisfied for all edges on every time slice. Here, $C_e(j)$ is the capacity of edge e on slice j. In all the experiments in this paper, each link capacity is assumed to be a constant across the time slices, i.e., $C_e(j) = C_e$ for all j.
- Expression (4) is the start and end time constraint for every job on every allowed path. Here, $I(S_i)$ and $I(E_i)$ are the time slices on which the requested start and end times (S_i and E_i, respectively) fall. The rate assignment must be zero before the start time and after the end time.

The number of variables required to solve the stage-1 problem is $\Theta(k \times |\mathcal{L}| \times |J|)$, where k is the maximum number of paths allowed for each job, $|\mathcal{L}|$ is the number of jobs in the network and $|J|$ is the number of time slices that need to be considered, which is determined by the maximum of the requested end times. Since stage 1 is a linear program, it can be solved in a reasonable amount of time for typical research

[1]First, list all the jobs in a sequence according to some administrative policy, priority, request times, or a combination of these. Next, conduct a binary search to find a last job in the sequence before which all the jobs can be finished ahead of their respective end times. The algorithms introduced in this paper can be used in each stage of the binary search.

[2]Note that the allowed paths can be time-varying. The resulting formulation can accommodate time-varying policy-induced constraints.

networks, as demonstrated in [20], [21]. We shall see that it is stage 2 that dominates the computation time, which must be coped with.

2) Stage 2: Note that the optimal solution that we obtain in stage 1 is not what the controller will apply to the network. The maximum concurrent throughput Z^* will be used in the constraints in stage 2 to achieve fairness among the jobs. In stage 2, we will find a near optimal "throughput" of the whole network while maintaining fairness, and we will obtain integer solutions for wavelength assignments.

For each job $i \in J$, following the convention in the MCF literature, we define the throughput for job i by

$$Z_i = \sum_{j \in \mathcal{L}} \sum_{p \in P(s_i,d_i,j)} x_i(p,j) \cdot LEN(j)/D_i. \quad (6)$$

The stage-2 optimization problem is as follows. It maximizes the weighted throughput of all jobs, where the weights are the (normalized) file sizes. The objective function gives preference to larger jobs. Implicitly, we regard larger jobs as being more important, since they are typically associated with large-scale scientific experiments, as shown by traffic studies on ESnet [8]. However, the network administrator can define other weight functions based on specific policy considerations. For instance, the weights can be inversely proportional to the file sizes. In that case, the resource allocation decision favors smaller jobs and tries to finish more smaller jobs at a possibly slight expense of larger jobs. Another possibility is that the users can specify how important their submitted jobs are. When the network is overloaded, these importance levels get translated into weights.

Stage 2

$$\max \quad \frac{\sum_{i \in J} Z_i D_i}{\sum_{i \in J} D_i} \quad (7)$$

$$\text{s.t.} \sum_{j=1}^{M} \sum_{p \in P(s_i,d_i,j)} x_i(p,j) LEN(j) = Z_i D_i, \quad \forall i \in J \quad (8)$$

$$Z_i \geq (1-\alpha) Z^*, \quad \forall i \in J \quad (9)$$

$$(3) - (4)$$

$$x_i(p,j) \in \mathbb{Z}_+, \forall p \in P(s_i,d_i,j), \forall i \in J, \forall j \in \mathcal{L}. \quad (10)$$

Some details about the state-2 formulation are as follows. First, we interpret C_e as the number of wavelengths available on link e; all the demands D_i are normalized by the capacity per wavelength; and $x_i(p,j)$ is the number of wavelengths assigned, which must be non-negative integers.

- Condition (8) says that, for each job, the total flow assignment, when summed over all time slices and allowed paths, must be equal to Z_i times the job size.
- Condition (9) is the fairness constraint. It ensures that the throughput of each job is at least some fraction of the maximum concurrent throughput Z^* computed in stage 1. Here, α is a parameter between 0 and 1.
- (3) and (4) are as the link capacity constraint and start and end time constraints as before.
- (10) is the integrality constraint for wavelength assignments $x_i(p,j)$. Here, $\mathbb{Z}_+$ denotes the set of non-negative integers.

Remark 1: The factor $1-\alpha$ in (9) increases the chance that the stage-2 problem has an integer solution for each $x_i(p,j)$. If not, α can be further increased.

Remark 2: In the overloaded situation where $Z_i < 1$ for some job i, Z_i tells how much job i should have its demand size reduced, for all i with $Z_i < 1$, in order to guarantee that the time constraints of all the jobs are met. If $Z_i > 1$ for some i, the solution $(x_i(p,j))_{p,j}$ can transfer up to Z_i times the demand size D_i. In the actual practice, we may assign any number of wavelengths between $\lceil x_i(p,j)/Z_i \rceil$ and $x_i(p,j)$ to job i on each path p and time slice j.

Heuristic Algorithm for the Stage-2 Problem - LPDAR

The computation time required to solve the stage-2 problem using standard integer programming solvers is prohibitively long. We have developed a much faster heuristic solution, which comes with the cost of a slight loss of optimality. In the heuristic solution, we start by ignoring the integrality constraint and treating the problem as a linear program. We then transform the continuous version of the solution into an integer version. The detailed steps are as follows.

1) We start by solving the stage-2 problem without the integrality constraint. That is, we first solve a linear program. The result will be called LP.
2) Given the real-valued solution to the linear program, we truncate the values for $x_i(p,j)$ down to the nearest integers. We call the result *Linear Programming-Discretized* (LPD).
3) After the truncation, there may be some remaining bandwidth (in terms of the number of wavelengths) left in the network. We next adjust the LPD solution by making use of the remaining bandwidth to get a better solution, which will be called *Linear Programming-Discretized with Adjusted Rates* (LPDAR). The adjustment is done by the greedy algorithm shown in Algorithm 1.

Algorithm 1 Greedy Algorithm for Bandwidth Adjustment

1: For each time slice $j \in \mathcal{L}$, for each job $i \in J$, and for each path $p \in P(s_i,d_i,j)$, find the remaining bandwidth of path p, RB_p, by:

$$RB_p \leftarrow \min_{e:e \in p} \{RB_e\}. \quad (11)$$

// *RB_e is the remaining bandwidth of edge e.*

2: Increase the bandwidth assignment on path p by RB_p:

$$x_i(p,j) \leftarrow x_i(p,j) + RB_p. \quad (12)$$

3: For each edge $e \in p$, decrease its remaining bandwidth by RB_p:

$$RB_e \leftarrow RB_e - RB_p, \quad \forall e \in p. \quad (13)$$

We next explain Algorithm 1. At each time slice j, for each job i and each of its allowed path p, the following takes place. The network controller finds the remaining bandwidth

on the path p, RB_p, which is the minimum of the remaining bandwidth of all the edges on the path p (see (11)). Then, we increase the bandwidth assignment $x_i(p,j)$ to use up all the remaining bandwidth on the path p (see (12)). After that, the remaining bandwidth of each edge $e \in p$ is decreased by RB_p (see (13)).

C. Relaxing End Times

In the algorithm presented in Section II-B, the end times of the jobs are strictly enforced, but the job sizes may be reduced depending on whether the network is overloaded or not. In some applications or for some users, it is more important to finish the entire transfer with a small amount but predictable delay than to enforce a strict deadline. In other words, if the network is overloaded, the deadlines can be extended but the the jobs must be transferred in their entirety. In this case, the network should try to find out the smallest possible end-time extension/relaxation. We call this the **Relaxing-End-Time (RET)** problem. After the network solves the RET problem, the new end times can be proposed to the network users, who may accept or decline the proposal. This negotiation process can be further repeated.

This section presents the algorithm for solving the RET problem. This algorithm is only needed when the network is overloaded and when some users are willing to relax the end times of their jobs. If the network is not overloaded, the algorithm in Section II-B suffices, which of course also handles the overloaded situation by asking the users to reduce the job sizes.

Our solution of the RET problem involves the following sub-problem, called **SUB-RET**.

SUB-RET

$$\min \quad \sum_{j\in\mathcal{L}} \gamma(j) \sum_{i\in J} \sum_{p\in P(s_i,d_i,j)} x_i(p,j) \tag{14}$$

$$\text{s.t.} \sum_{j\in\mathcal{L}} \sum_{p\in P(s_i,d_i,j)} x_i(p,j)\cdot LEN(j) \geq D_i, \forall i \in J \tag{15}$$

$$x_i(p,j) = 0,\ \forall p \in P(s_i,d_i,j), \quad j \leq I(S_i) \text{ or } j > I((1+b)E_i), \forall i \in J \tag{16}$$

$$(3), (10).$$

SUB-RET should be understood primarily as a feasibility problem; the objective function (14) is not essential and can be substituted by other functions. We introduce a parameter b in (16) to measure how much the end times should be extended; $(1+b)$ is the factor by which each of the end times is relaxed. Under a particular b, any feasible solution to SUB-RET can complete every job i by the time $I((1+b)E_i)$, where $I(\cdot)$ is the rounding of the time on slice boundaries. Later, in Algorithm 2, we will find the smallest b for which SUB-RET is feasible.

We next explain some details about SUB-RET.

- $\mathcal{L}$ is understood to contain all the relevant time slices after extending the end times by a factor $(1+b)$.
- Expression (14) is known as the **Quick-Finish (QF)** objective function, which was first introduced in [21]. In this function, $\gamma(j)$ is a given cost function increasing in time j. We choose it to be $\gamma(j) = j+1$ in our evaluations. The intention is to finish the jobs earlier rather than later if possible. The solution tends to pack more flows in earlier time slices, but leaves the network load light to better accommodate future job requests.
- Expression (15) says the sum of total flows assigned on all time slices and on all allowed paths for job i is greater than or equal to the demand of job i. That is, each job i has to be completed by the extended end time.
- Expression (16) ensures the flow assignment can be positive from the start time to the extended end time.
- (3) is the link capacity constraint; (10) is the non-negative integer wavelength assignment requirement.

Remark: An alternative way to extend the deadline is by extending the intervals between the start and end times by the same constant factor $1+b$ for all jobs. For brevity, we do not further discuss this option in this paper.

There are two remaining questions: how to find the smallest value b for which the SUB-RET problem is feasible and how to cope with the integrality constraints. For the first question, we use a binary search method to find the smallest b. For the second question, we use the same heuristic solution LPDAR in Section II. To guarantee that all the jobs will finish in the end using LPDAR, we add another loop into the algorithm that further extends the end times a bit more if the network still cannot satisfy all the jobs' requirement. The final algorithm is listed as Algorithm 2. In the algorithm, b_{max} is a given large enough constant; δ is a small constant, e.g., $\delta = 0.1$ in our evaluations.

Algorithm 2 RET

1: Perform binary search on $[0, b_{max}]$ to find the minimum b for which the SUB-RET problem is feasible *without the integrality constraints* (i.e., replacing (10) by (5) in SUB-RET). Call the minimum $\hat{b}$.
2: Obtain the real-valued solution to SUB-RET under $\hat{b}$ without the integrality constraints; apply Algorithm 1 to get the integer-valued solution.
3: **if** the bandwidth assignment results generated by step 2 cannot satisfy the requirement of all the jobs **then**
4: $\quad \hat{b} \leftarrow \hat{b} + \delta$; go back to step 2.
5: **end if**

III. Evaluation

The performance improvement of the overall optimization framework with multipath, time-varying bandwidth allocation and periodical reallocation has been verified previously [21]. Here, we focus on performance evaluation of the algorithms introduced in the paper. The focus is to examine the performance and speed of the sub-algorithm, LPDAR, which is a heuristic technique to find an integer solution based on the initial linear programming solution. LPDAR is the key component in both algorithms for solving the maximizing-throughput problem and the RET problem.

We will compare LPDAR to LP and LPD. Since it is practically impossible to get the optimal integer solutions using standard solvers for mixed integer programming but for very

small setups, we will not be able to show those results. Instead, we use the LP solutions as performance benchmarks, since they generally provide upper bounds for the optimal integer solutions. However, LP does not produce integer solutions in general. We also evaluate the required computation time to determine the scalability of the algorithms, and will show that both LPDAR and LPD are fast algorithms. But, LPDAR produces much better results than the less sophisticated LPD.

The experiments were conducted on random networks and the Abilene network. The latter was the backbone network for Internet 2. Experiments on the Abilene network show how well the proposed algorithms work on practical networks; experiments on the random networks show how general the observed results are or how they scale with the network size. In this paper, we show only a subset of the results, which are typical instances. The random networks that we use typically have between 100 to 400 nodes, with an average node degree of 4. We use the network generator, BPRITE [28], to generate the random networks using Waxman's algorithm [29]. In Waxman's algorithm, nodes are placed on a plane; the probability of interconnecting two nodes decreases exponentially with the Euclidean distance between them. Each link has a capacity of 20 Gbps. Our instance of the Abilene network consists of a backbone of 11 nodes. The backbone links each have 20 Gbps capacity. The job size is uniformly distributed between $[1, 100]$ Gigabytes. The α value in the stage-2 problem is set to 0.1. We use the commercial CPLEX package for solving linear programming problems on Intel-based workstations.

A. Maximizing Throughput with End Time Guarantee

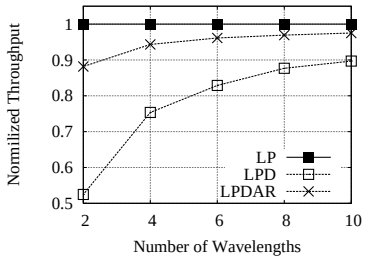

Fig. 1. Throughput comparison of LP, LPD and LPDAR on a random network with 100 nodes and 200 pairs of links

1) Throughput Comparison: : Fig. 1 shows that the throughput comparison of LP, LPD and LPDAR under different numbers of wavelengths on each link while holding the capacity of each link constant. The throughput is normalized with respect to that of LP. The network has 100 nodes and 200 pairs of links. LP achieves the best possible throughput but does not satisfy the integrality constraint. From Fig. 1, we see that the throughput of LPD suffers when the number of wavelengths per link is small. When the network has only two wavelengths on each link, LPD achieves about half of throughput of LP. The reason is that, when the number of wavelengths is small, LPD tends to truncate a non-trivial portion of the continuous bandwidth assignment in order to comply with the integer-valued wavelength constraint. LPDAR, which adjusts the wavelength assignment by utilizing the remaining bandwidth on the paths when possible, performs much better than LPD. For instance, LPDAR achieves nearly 90% of the throughput of LP in the case of two wavelengths per link; it achieves 95% or more of the throughput of LP with four or more wavelengths per link. In short, the simple greedy algorithm of LPDAR is effective for throughput improvement.

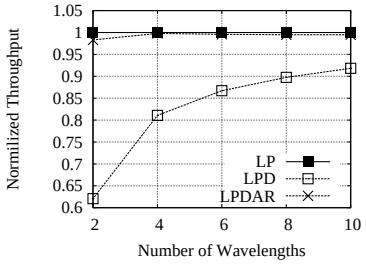

Fig. 2. Throughput comparison between LP, LPD and LPDAR on the Abilene network with 11 nodes and 20 pairs of links

Fig. 2 shows the normalized throughput of LP, LPD, and LPDAR in the Abilene network with 11 backbone nodes and 20 pairs of links. We see that LPD still suffers much throughput reduction when the number of wavelengths per link is small. At two wavelengths per link, LPD achieves around 60% of the throughput of LP. The throughput improvement when using LPDAR is more dramatic than in the random network case. LPDAR achieves nearly identical throughput as LP.

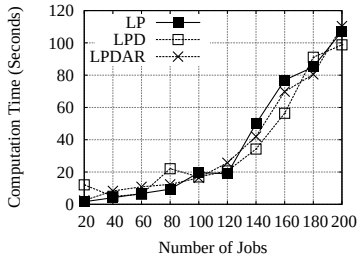

Fig. 3. Computation time comparison of LP, LPD and LPDAR on a random network with 100 nodes and 200 pairs of links.

2) Computation Time: Recall that the motivation to develop the heuristic algorithm, LPDAR, is that it is impractical to solve the original integer programming problem of stage 2 using standard solvers. Fig. 3 shows representative results for comparing the computation times of LP, LPD and LPDAR. We have not been able to measure the computation time required for solving the optimal integer version since this takes too long. Note that the computation times of the three algorithms are quite similar. This is because LPD and LPDAR also need to solve the linear program first and this step dominates the computation time. In our previous papers [20], [21], we conducted many experiments on the computation time under different parameters and showed that linear programming formulations can be solved quickly enough for practical scenarios. In light

of the results in Fig. 3, that conclusion carries over to our current situation.

B. Relaxing End Times

In the RET problem, completing all the jobs is more important than strictly enforcing the end times. In this part, we compare the fraction of jobs finished and average end time in the solutions obtained by LP, LPD and LPDAR at the end of Algorithm 2.

1) Fraction of jobs finished: We computed the fraction of jobs finished by LP, LPD and LPDAR for a variety of scenarios. Algorithm 2 guarantees to find a $\hat{b}$ such that, when the end times are extended by $(1+\hat{b})$, the solution by LPDAR (and hence, by LP) can complete all the jobs. In contrast, under the same extended end times, LPD only finished a very small fraction (typically zero) of the jobs for these cases. Furthermore, the value of $\hat{b}$ was often the same as (or slightly larger than) the minimum value under which the LP solution can finish all the jobs. In short, LPDAR has performance comparable to LP but substantially better than LPD.

2) Average End Time: The average end time of all jobs is a measure of how fast the jobs are completed. Fig. 4 compares the average end time of the results by LP and LPDAR. The unit is in the number of time slices. Note that most of the jobs in the solution of LPD are not finished by the end of Algorithm 2 execution. Hence, the average end time is not relevant in the LPD case.

In these experiments, we use the Quick-Finish objective function for the SUB-RET problem, which has the effect of trying to finish a job earlier if possible. As expected, LP has a smaller average end time since it does not have the integrality constraints. LPDAR is nearly as good as LP, despite that it gives integer solutions. Similar results have been observed on the Abilene network, which are omitted.

Fig. 4 shows that the average end time increases as the number of jobs increases. This is because the network does not change in the experiments while the number of jobs varies.

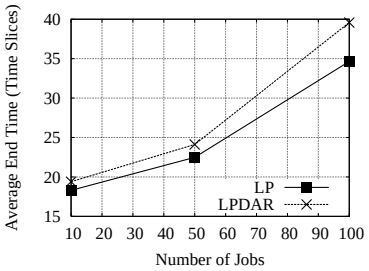

Fig. 4. Average end time comparison of LP, LPD and LPDAR on a random network with 100 nodes and 200 pairs of links

IV. Additional Related Work

Compared with the traditional QoS frameworks, such as DiffServ [30], the ATM network [31], or MPLS [32], admission control and scheduling for research networks are recent concerns with much fewer published studies.

The NSF-supported DRAGON [13] project develops control plane architecture and middleware for multi-domain traffic engineering and resource allocation, e.g., using GMPLS protocols [33] for setting up SONET circuits or lightpaths. It supports advance reservations of label switched paths (LSP) on requested time periods. CHEETAH [15] is a similar project to DRAGON but it focuses on simpler, distributed operations for path computation and bandwidth management to support high arrival rates of immediate connection requests. OSCARS [16] is the control plane project for DOE's ESnet, also similar to DRAGON. BRUW is the counterpart of OSCARS for Internet2. Other notable related work in this category includes [12], [14], [34] [35], [36]. Most of the above control-plane architectures and tools provide rudimentary AC and scheduling algorithms for simple job types.

Recent papers on AC and scheduling algorithms for bulk transfers with advance reservations include [22], [17], [20], [21], [23], [19], [24], [25]. All these papers study the problem for router networks where the bandwidth is assumed to be infinitely divisible. In [19], the AC and scheduling problem is considered only for the single link case. As a result, multi-path routing and network-level bandwidth allocation are not relevant issues in [19]. The problem studied in [22] is similar to those examined in our earlier papers [20], [21], but the emphasis is different.In [23], the authors consider single-link admission control or link-by-link admission control under single-path routing. The admission control does not rely on formulating optimization problems. The AC decision is based on the job's average bandwidth requirement, which is computed using the size of the job and the deadline information. The bandwidth of existing jobs may be re-allocated in the single link case but not in the network case. The authors of [25] propose a malleable reservation scheme for bulk transfer, which checks every possible interval between the requested start and end times for the job and tries to find a path that can accommodate the entire job on that interval. In [24], the computational complexity of a related path-finding problem is studied and an approximation algorithm is suggested. [17] starts with an advance reservation problem for bulk transfer, but converts it into a constant bandwidth allocation problem to maximize the job acceptance rate. The bandwidth constraints are at the ingress and egress links only, and hence, there is no routing issue.

Several earlier studies [37], [38], [39] have considered admission control at an individual link for applications that require minimum bandwidth guarantee on some time intervals. The concern is typically about designing efficient data structures[38]. [24], [40], [25], [41] and [39] go beyond single-link advance reservations and tackle the more general path-finding problem, but typically only for new requests, one at a time. The routes and bandwidth of existing jobs are unchanged. The authors of [18] advocate periodic re-optimization to determine new bandwidth allocation. However, they do not assume that users make advance reservations with requested end times. Many papers study advance reservations, re-routing, or re-optimization of lightpaths in wavelength-based optical networks [26], [27]. But, they do not consider the start and end time constraints and do not focus on bulk transfers.

V. Conclusion

In this paper, we present two algorithms for scheduling time-constrained bulk transfers on wavelength-based optical research networks. The first one works on both underloaded or overloaded networks. In the overloaded case, the algorithm seeks to decrease the job sizes in a way that maximizes the network throughput while maintaining a level of fairness among the jobs. The second algorithm works in the overloaded case and serves as an alternative to the first algorithm. It seeks to extend the end times by the smallest possible proportion and complete all the jobs by the extended end times.

The main challenge for us is that the underlying problems are integer optimization problems for wavelength assignment, which have no known fast optimal solutions. We present a heuristic sub-algorithm called LPDAR, which converts fractional solutions from linear programming into integer solutions. LPDAR is the key component used in both aforementioned algorithms. Evaluation shows that LPDAR leads to very good algorithms with a performance level and speed both comparable to those of the LP fractional solutions.

References

[1] The U.K. Research Councils, http://www.research-councils.ac.uk/escience/, site last visited on Feb. 18, 2008.

[2] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[3] Internet2, http://www.internet2.edu, site last visited on Feb. 18, 2008.

[4] Energy Science Network (ESnet), http://www.es.net.

[5] National Lambda Rail, http://www.nlr.net, site last visited on Feb. 18, 2008.

[6] CA*net4, http://www.canarie.ca/canet4/index.html, site last visited on Feb. 18, 2008.

[7] GÉANT2, http://www.geant2.net/.

[8] W. E. Johnston, J. Metzger, M. OConnor, M. Collins, J. Burrescia, E. Dart, J. Gagliardi, C. Guok, and K. Oberman, "Network communication as a service-oriented capability," in *High Performance Computing and Grids in Action, Vol. 16*, L. Grandinetti, Ed. IOS Press, 2008, http://www.es.net/pub/esnet-doc/ESnet-SciDAC-Review-Summer-2007.pdf.

[9] H. B. Newman, M. H. Ellisman, and J. A. Orcutt, "Data-intensive e-science frontier research," *Communications of the ACM*, vol. 46, no. 11, pp. 68–77, Nov. 2003.

[10] J. Bunn and H. Newman, "Data-intensive grids for high-energy physics," in *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and T. Hey, Eds. John Wiley & Sons, Inc, 2003.

[11] T. Ferrari, *Grid Network Services Use Cases from the e-Science Community*, The Open Grid Forum, Dec. 2007, http://www.ogf.org/Public_Comment_Docs/Documents/Jul-2007/draft-ggf-ghpn-netservices-usecases-2-12.pdf, site last visited on Feb. 18, 2008.

[12] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservations and co-allocation," in *Proceedings of the International Workshop on Quality of Service (IWQoS '99)*, 1999.

[13] T. Lehman, J. Sobieski, and B. Jabbari, "DRAGON: A framework for service provisioning in heterogeneous grid networks," *IEEE Communications Magazine*, March 2006.

[14] Hybrid Optical and Packet Infrastructure, http://networks.internet2.edu/hopi.

[15] CHEETAH: Circuit-switched High-speed End-to-End Transport Architecture, http://www.ece.virginia.edu/cheetah/.

[16] OSCARS: On-demand Secure Circuits and Advance Reservation System, http://www.es.net/oscars.

[17] L. Marchal, P. Primet, Y. Robert, and J. Zeng, "Optimal bandwidth sharing in grid environment," in *Proceedings of IEEE High Performance Distributed Computing (HPDC)*, June 2006.

[18] R. Bhatia, M. Kodialam, and T. V. Lakshman, "Fast network re-optimization schemes for MPLS and optical networks," *Computer Networks*, vol. 50, no. 3, Feb. 2006.

[19] S. Naiksatam and S. Figueira, "Elastic reservations for efficient bandwidth utilization in LambdaGrids," *The International Journal of Grid Computing*, vol. 23, no. 1, pp. 1–22, January 2007.

[20] K. Rajah, S. Ranka, and Y. Xia, "Scheduling bulk file transfers with start and end times," *Computer Networks*, vol. 52, no. 5, pp. 1105–1122, April 2008, .

[21] ——, "Advance reservation and scheduling for bulk transfers in research networks," *IEEE Transactions on Parallel and Distributed Systems, Accepted*, .

[22] B. B. Chen and P. V.-B. Primet, "Scheduling deadline-constrained bulk data transfers to minimize network congestion," in *Proceedings of the Seventh IEEE International Symposium on CCGRID*, May 2007.

[23] K. Munir, S. Javed, M. Welzl, H. Ehsan, and T. Javed, "An end-to-end QoS mechanism for grid bulk data transfer for supporting virtualization," in *Proceedings of IEEE/IFIP International Workshop on End-to-end Virtualization and Grid Management (EVGM 2007)*, San Jose, California, October 2007.

[24] R. Guerin and A. Orda, "Networks with advance reservations: The routing perspective," in *Proceedings of IEEE INFOCOM 17*, 1999.

[25] L.-O. Burchard and H.-U. Heiss, "Performance issues of bandwidth reservation for grid computing," in *Proceedings of the 15th Symposium on Computer Archetecture and High Performance Computing (SBAC-PAD'03)*, 2003.

[26] D. Banerjee and B. Mukherjee, "Wavelength-routed optical networks: linear formulation, resource budgeting tradeoffs, and a reconfiguration study," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 598–607, Oct. 2000.

[27] E. Bouillet, J.-F. Labourdette, R. Ramamurthy, and S. Chaudhuri, "Lightpath re-optimization in mesh optical networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 437–447, 2005.

[28] A. Medina, A. Lakhina, I. Matta, and J. Byers, QoS Networking Laboratory (QNL), Boston University, http://www.cs.bu.edu/brite/, site last visited on Feb. 11, 2009.

[29] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Select Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, December 1988.

[30] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An architecture for differentiated services*, RFC 2475, IETF, Dec. 1998.

[31] D. E. McDysan and D. L. Spohn, *ATM Theory and Applications*. McGraw-Hill, 1998.

[32] E. Rosen, A. Viswanathan, and R. Callon, *Multiprotocol label switching architecture*, RFC 3031, IETF, Jan. 2001.

[33] E. Mannie, *Generalized multi-protocol label switching (GMPLS) architecture*, RFC 3945, IETF, Oct. 2004.

[34] E. He, X. Wang, V. Vishwanath, and J. Leigh, "AR-PIN/PDC: Flexible advance reservation of intradomain and interdomain lightpaths," in *Proceedings of the IEEE GLOBECOM 2006*, 2006.

[35] C. Curti, T. Ferrari, L. Gommans, B. van Oudenaarde, E. Ronchieri, F. Giacomini, and C. Vistoli, "On advance reservation of heterogeneous network paths," *Future Generation Computer Systems*, vol. 21, no. 4, pp. 525–538, Apr. 2005.

[36] R. Boutaba, W. Golab, Y. Iraqi, T. Li, and B. Arnaud, "Grid-controlled lightpaths for high performance grid applications," *Journal of Grid Computing*, vol. 1, no. 4, pp. 387–394, December 2003.

[37] O. Schelén, A. Nilsson, J. Norrgård, and S. Pink, "Performance of QoS agents for provisioning network resources," in *Proceedings of IFIP Seventh International Workshop on Quality of Service (IWQoS'99)*, London, UK, June 1999.

[38] A. Brodnik and A. Nilsson, "A static data structure for discrete advance bandwidth reservations on the Internet," Department of Computer Science and Electrical Engineering, Luleå University of Technology, Sweden, Tech. Rep. Tech report cs.DS/0308041, 2003.

[39] L.-O. Burchard, J. Schneider, and B. Linnert, "Rerouting strategies for networks with advance reservations," in *Proceedings of the First IEEE International Conference on e-Science and Grid Computing (e-Science 2005)*, Melbourne, Australia, Dec. 2005.

[40] T. Wang and J. Chen, "Bandwidth tree - A data structure for routing in networks with advanced reservations," in *Proceedings of the IEEE International Performance, Computing and Communications Conference (IPCCC 2002)*, April 2002.

[41] T. Erlebach, "Call admission control for advance reservation requests with alternatives," Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology (ETH) Zurich, Tech. Rep. TIK-Report Nr. 142, 2002.